PennState

CSE543
Introduction to Computer and Network Security
Module: Authentication

Asst. Prof. Syed Rafiul Hussain

Kerberos

PennState

- History: from UNIX to Networks (late 80s)
  - Solves: password eavesdropping
    - Also mutual authentication
  - Online authentication
    - Variant of Needham-Schroeder protocol
  - Easy application integration API
  - First *single sign-on system* (SSO)
  - Genesis: rsh, rcp
    - authentication via assertion

- Most widely used (non-web) centralized password system in existence (and lately only one…)
- Now: Windows 2K/XP/Vista/etc network authentication
  - Old Windows authentication was a cruel joke.

An aside …

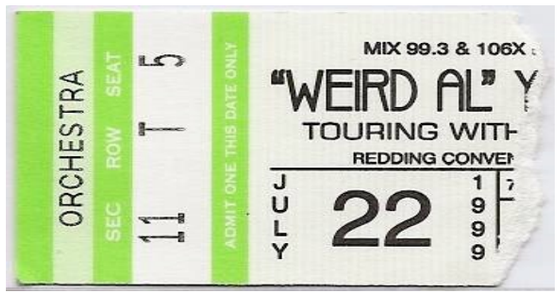
- Authentication
  - Assessing identity of users
  - By using credentials …
- Authorization
  - Determining if users have the right to perform requested action (e.g., write a file, query a database, etc.)

- Kerberos authenticates users, but does not perform any authorization functions …
  - … beyond identify user as part of Realm
  - Typically done by application.

- Q: Do you use any "*Kerberized*" programs?
  - How do you know?

The setup …

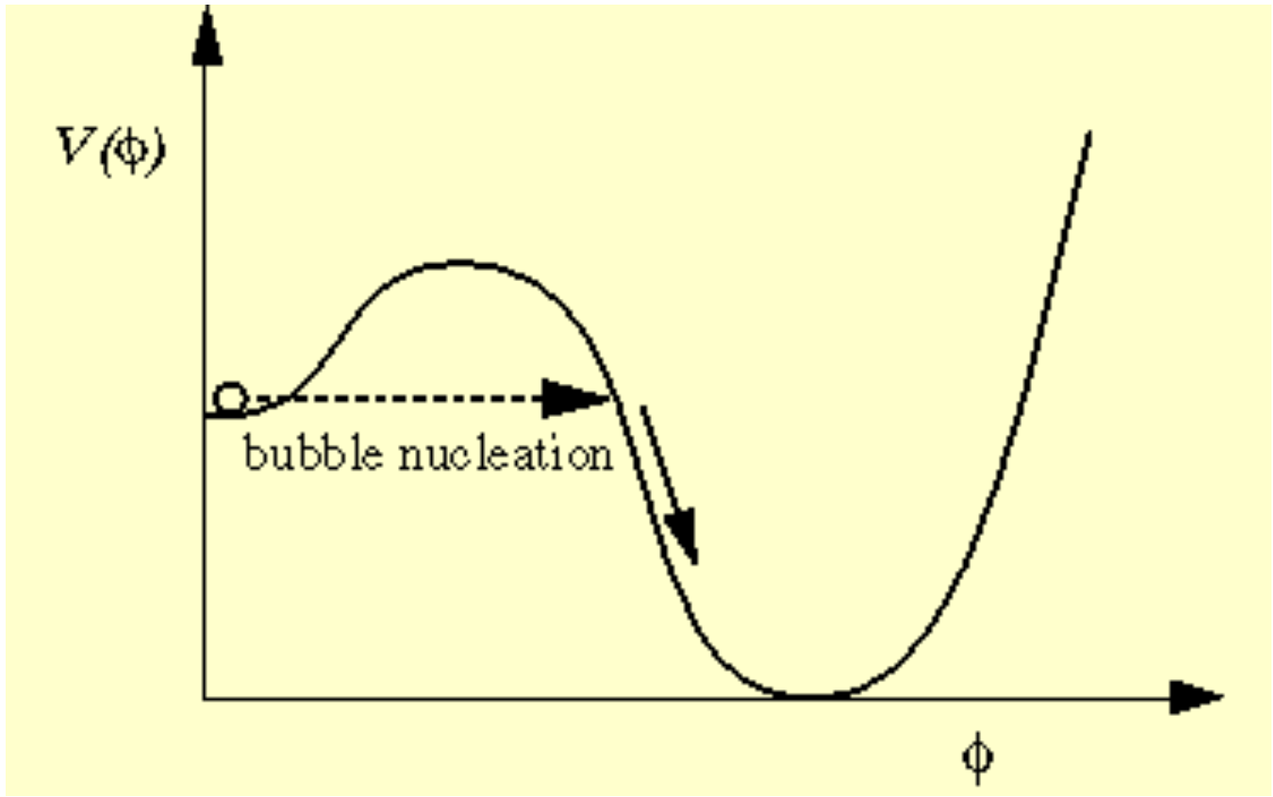
- The players
  - Principal - person being authenticated
  - Service (verifier) - entity requiring authentication (e.g, AFS)
  - Key Distribution Center (KDC)
    - Trusted third party for key distribution
    - Each principal and service has a Kerberos password known to KDC, which is munged to make a password ke, e.g., k^A
  - Ticket granting server
    - Server granting transient authentication



- The objectives
  - Authenticate Alice (Principal) to Bob (Service)
  - Negotiate a symmetric (secret) session key k^{AB}

The protocol

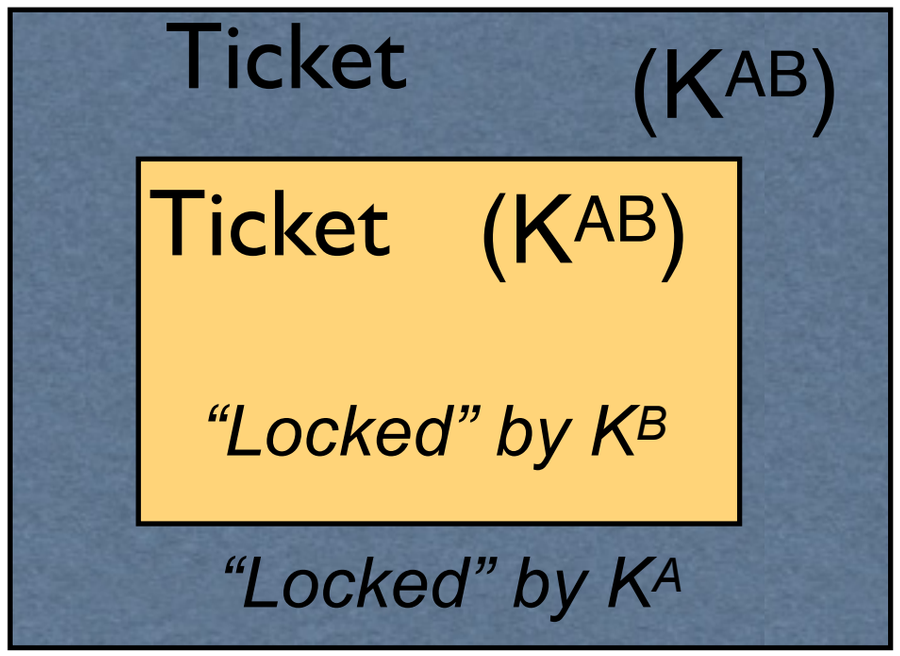
- A two-phase process
  1. User authentication/obtain session key (and ticket granting ticket) key from Key Distribution Center
  2. Authenticate Service/obtain session key for communication with service



- Setup
  - Every user and service get certified and assigns password

A Kerberos Ticket

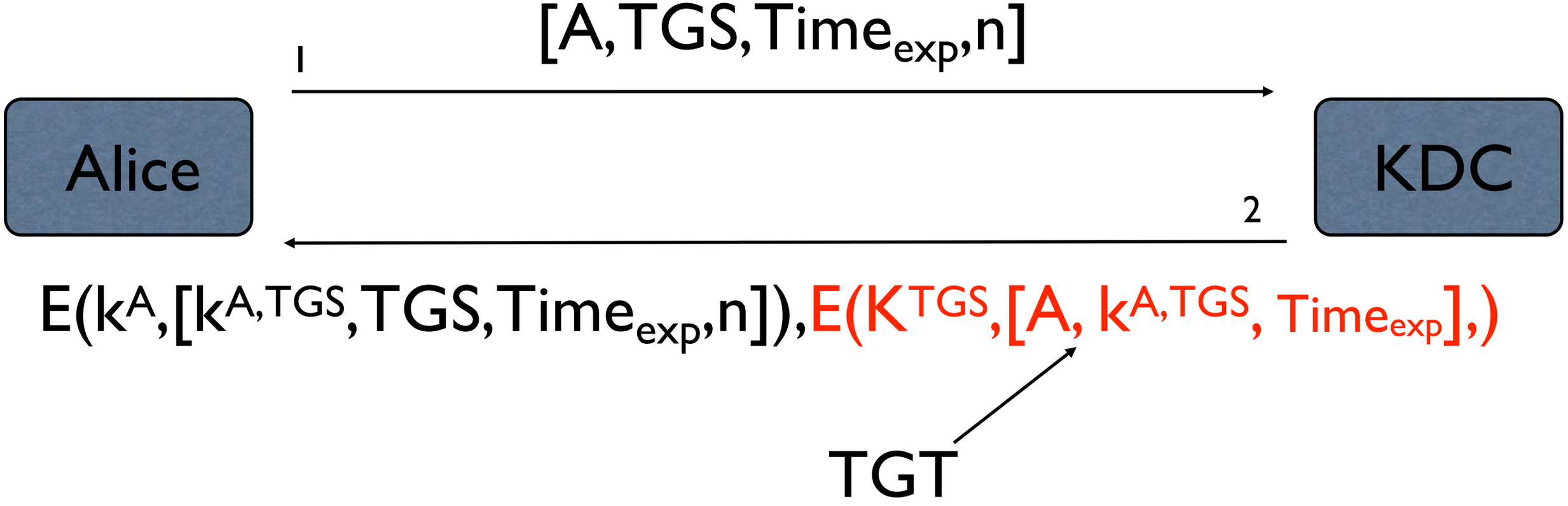
- A kerberos ticket is a token that …
  - Alice is the only one that can open it
  - Contains a session key for Alice/Bob (K^{AB})
  - Contains *inside it* a token that can only be opened by Bob
- Bob's Ticket contains
  - Alice's identity
  - The session key (K^{AB})



- Q: What if issuing service is not trusted?

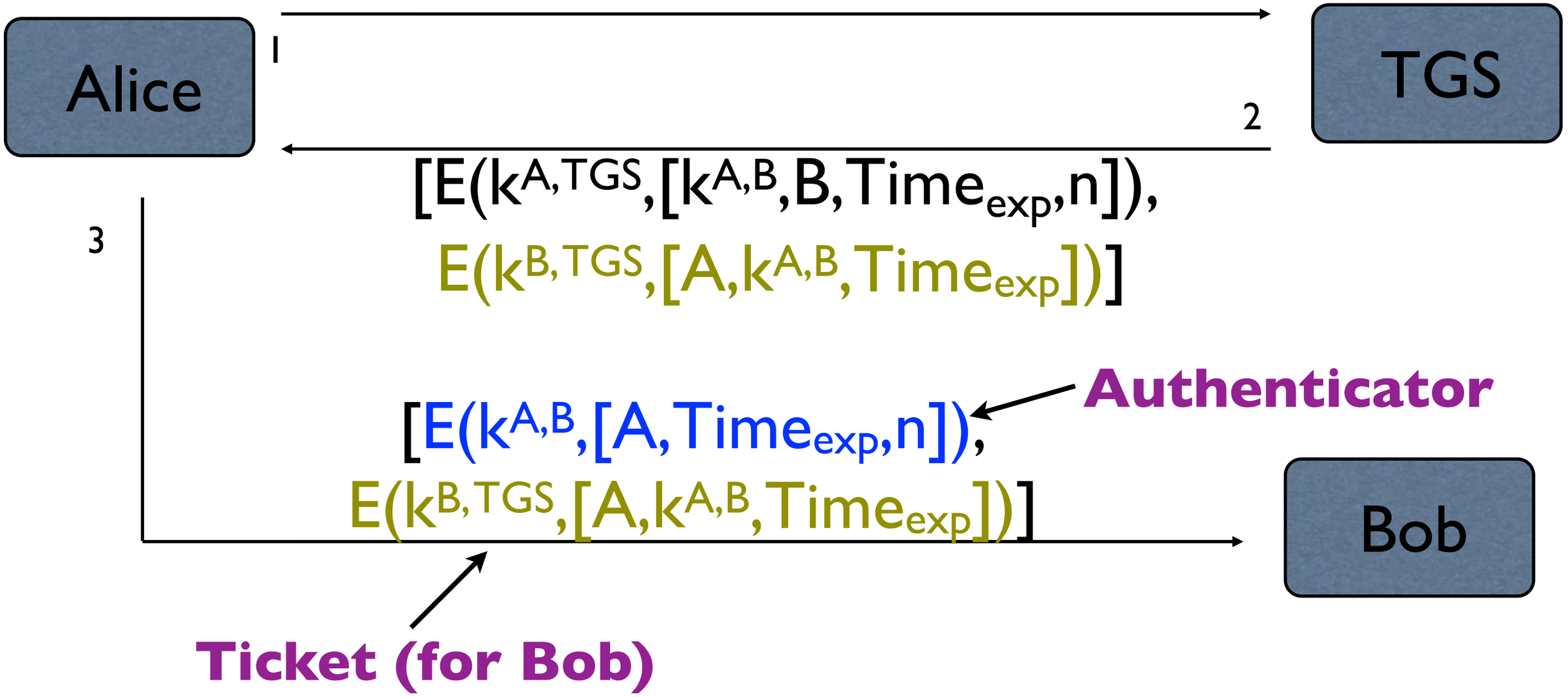
Phase 1 (obtaining a TGT)

- $Time_{exp}$ - time of expiration
- n - nonce (random, one-use value: e.g., timestamp)



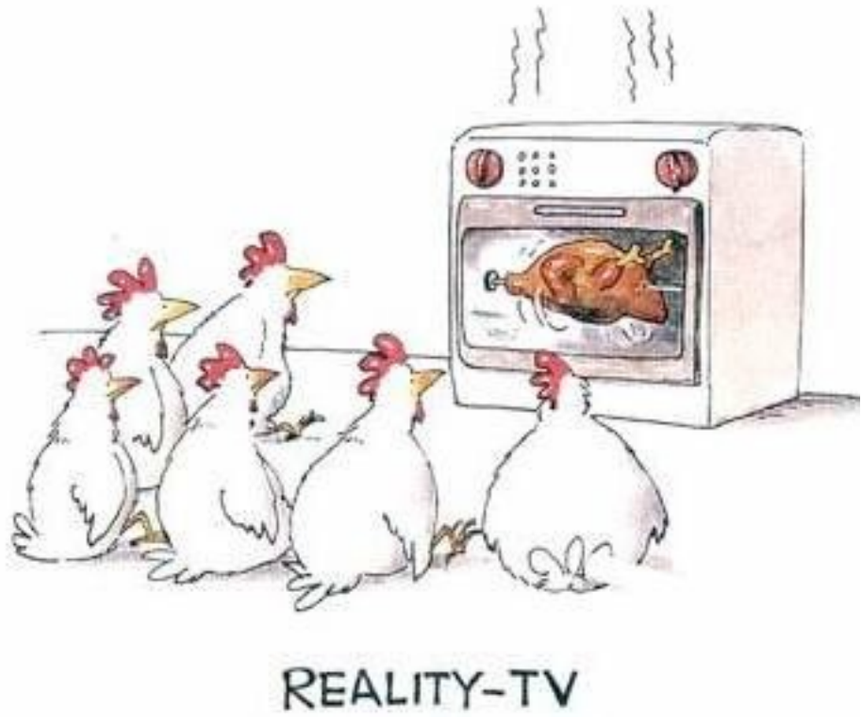
Phase 2 (authentication/key dist.)

PennState



Kerberos Reality

- V4 was supposed to be replaced by V5
  - But wasn't because interface was ugly, complicated, and encoding was infuriating
- Assumes *trusted path* between user and Kerberos
- Widely used in UNIX domains
- Robust and stable implementation

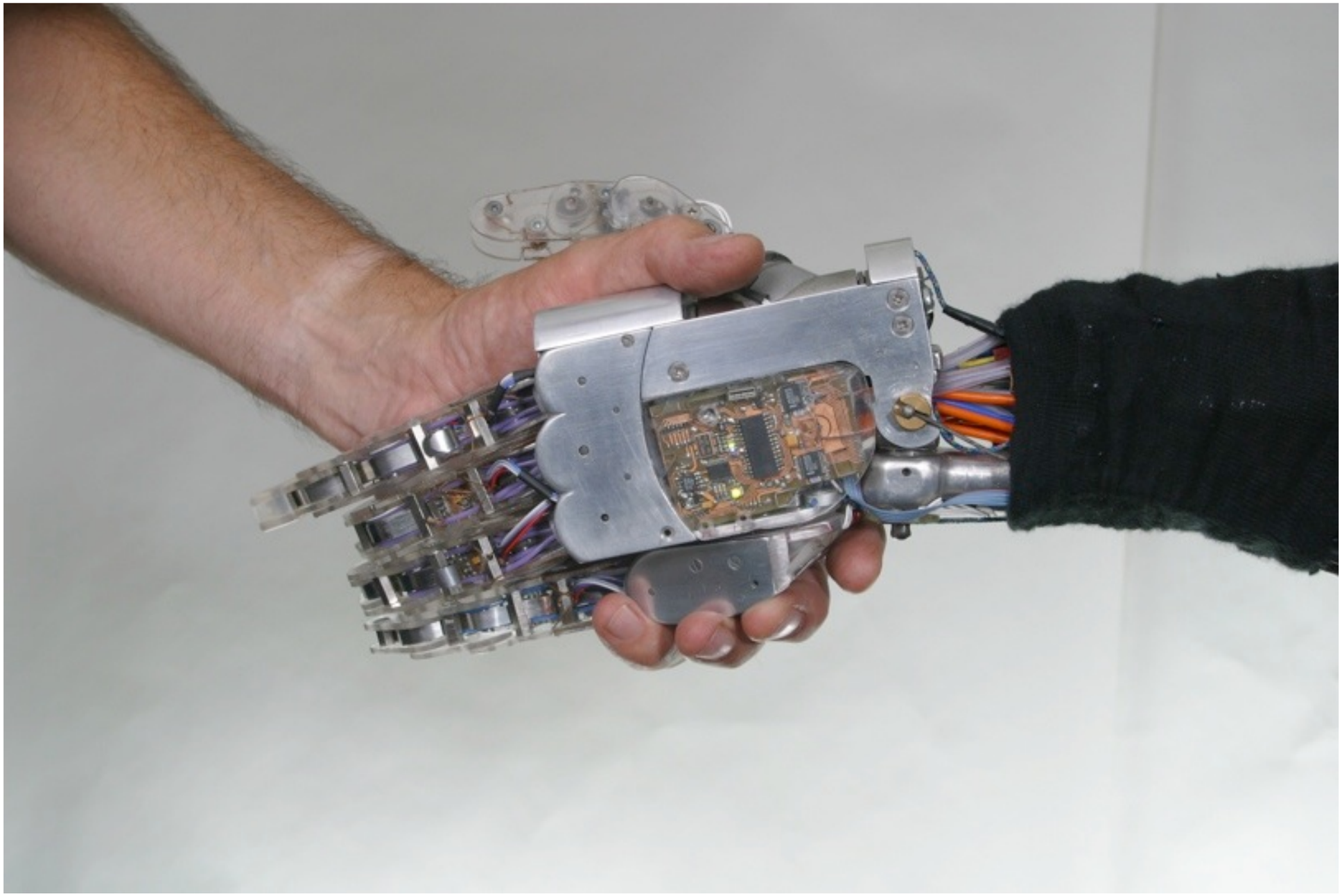


- *Problem*: trust ain't transitive, so not so good for large collections of autonomous enterprises

Meeting Someone New

PennState

- Anywhere in the Internet

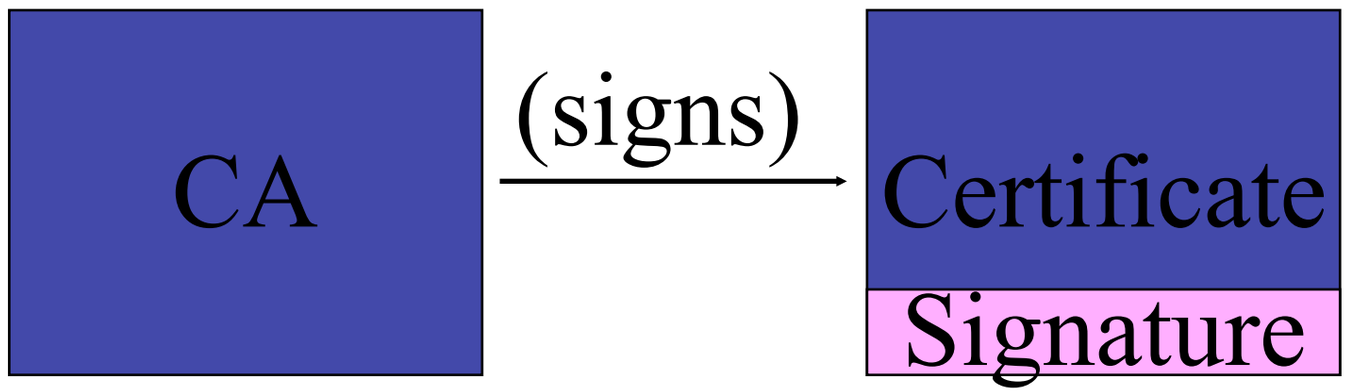


What is a certificate?

- A certificate …
  - … makes an association between a user identity/job/attribute and a private key
  - … contains public key information {e,n}
  - … has a validity period
  - … is signed by some *certificate authority* (CA)
  - ... identity may have been vetted by a *registration authority* (RA)
- Issued by CA for some purpose
  - Symantec is in the business of issuing certificates
  - People trust Symantec (formerly Verisign) vet identity

Why do I trust the certificate?

- A collections of "root" CA certificates
  - … baked into your browser
  - … vetted by the browser manufacturer
  - … supposedly closely guarded (yeah, right)
- Root certificates used to validate certificate
  - Vouches for certificate's authenticity



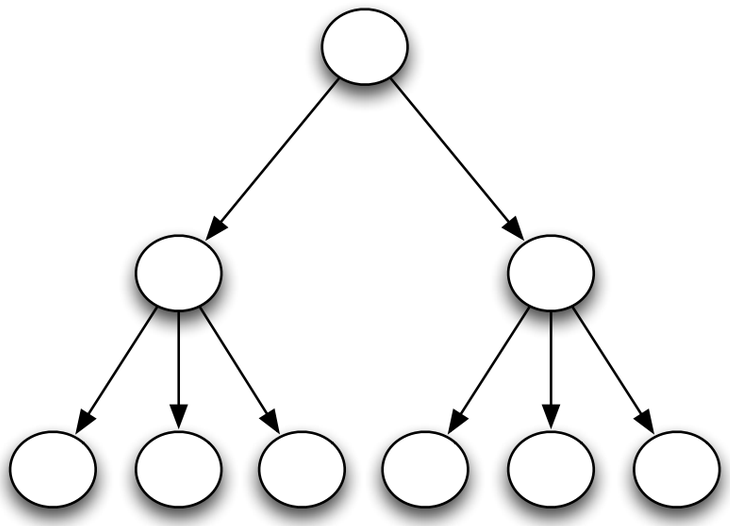
Public Key Infrastructure

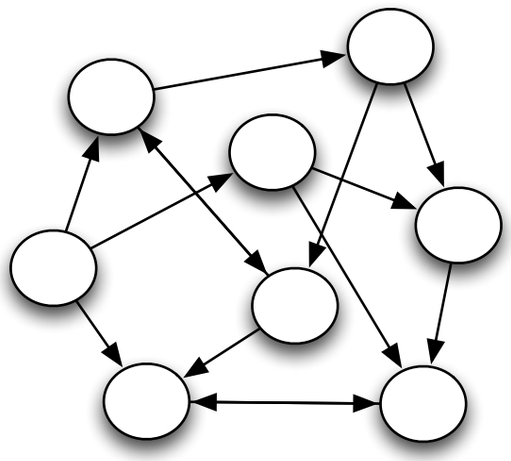
- System to "*securely distribute public keys (certificates)*"
  - Q: Why is that hard?

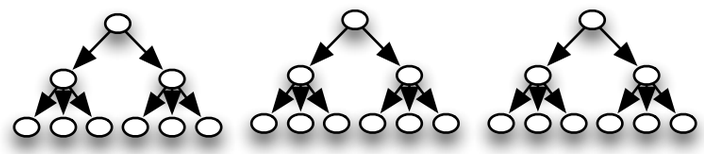
- Terminology:
  - Alice signs a certificate for Bob's name and key
    - Alice is issuer, and Bob is subject
  - Alice wants to find a path to Bob's key
    - Alice is verifier, and Bob is target
  - Anything that has a public key is a principal
  - Anything trusted to sign certificates is a trust anchor
    - Its certificate is a root certificate

Possible PKI Constructions

- Monarchy
  - Single globally trusted third party
- Anarchy
  - No globally trusted third party
    - e.g., Using MIT's PGP keyserver
- Oligarchy
  - Multiple globally trusted third parties
    - Model used in the Internet

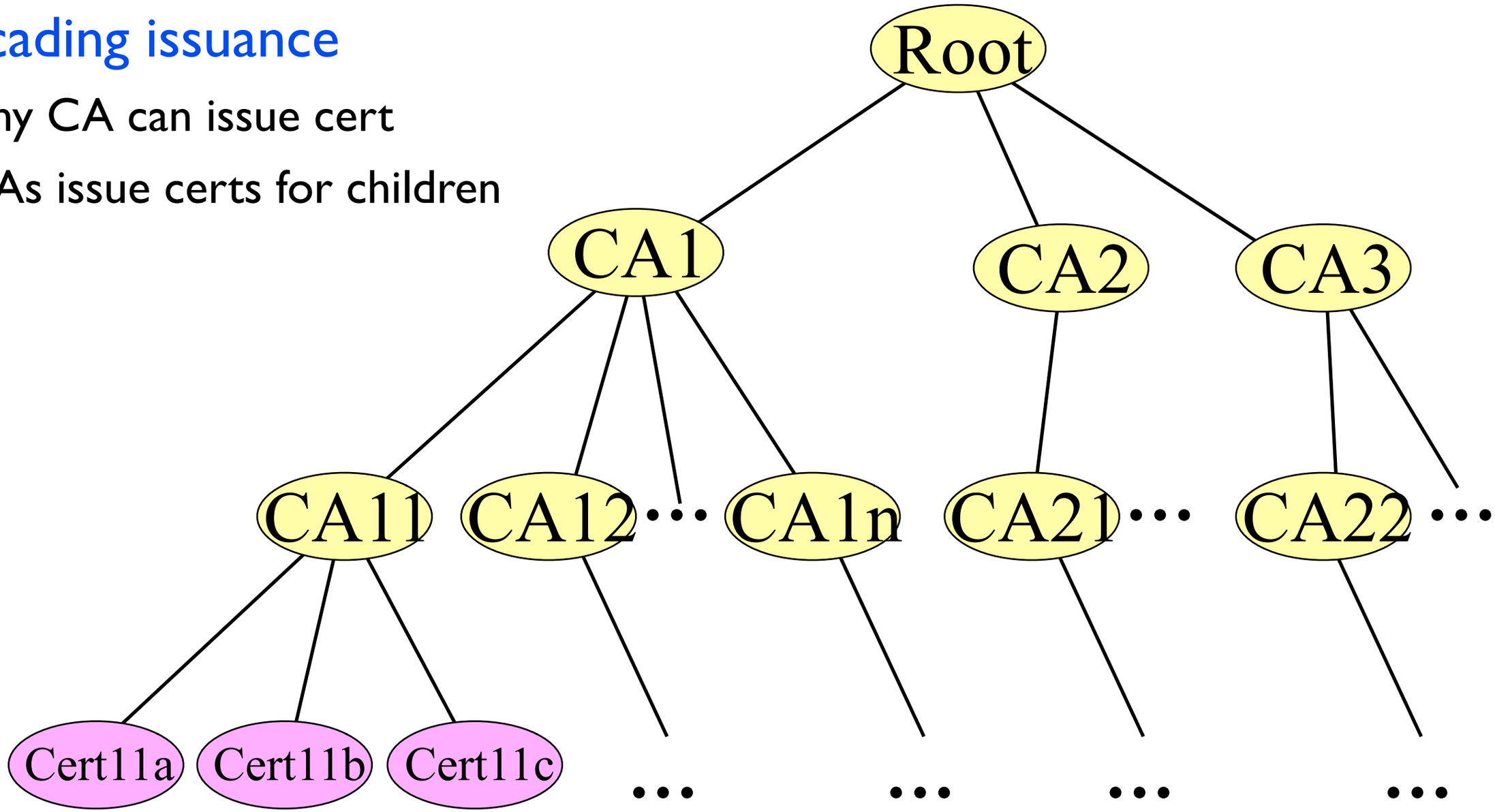




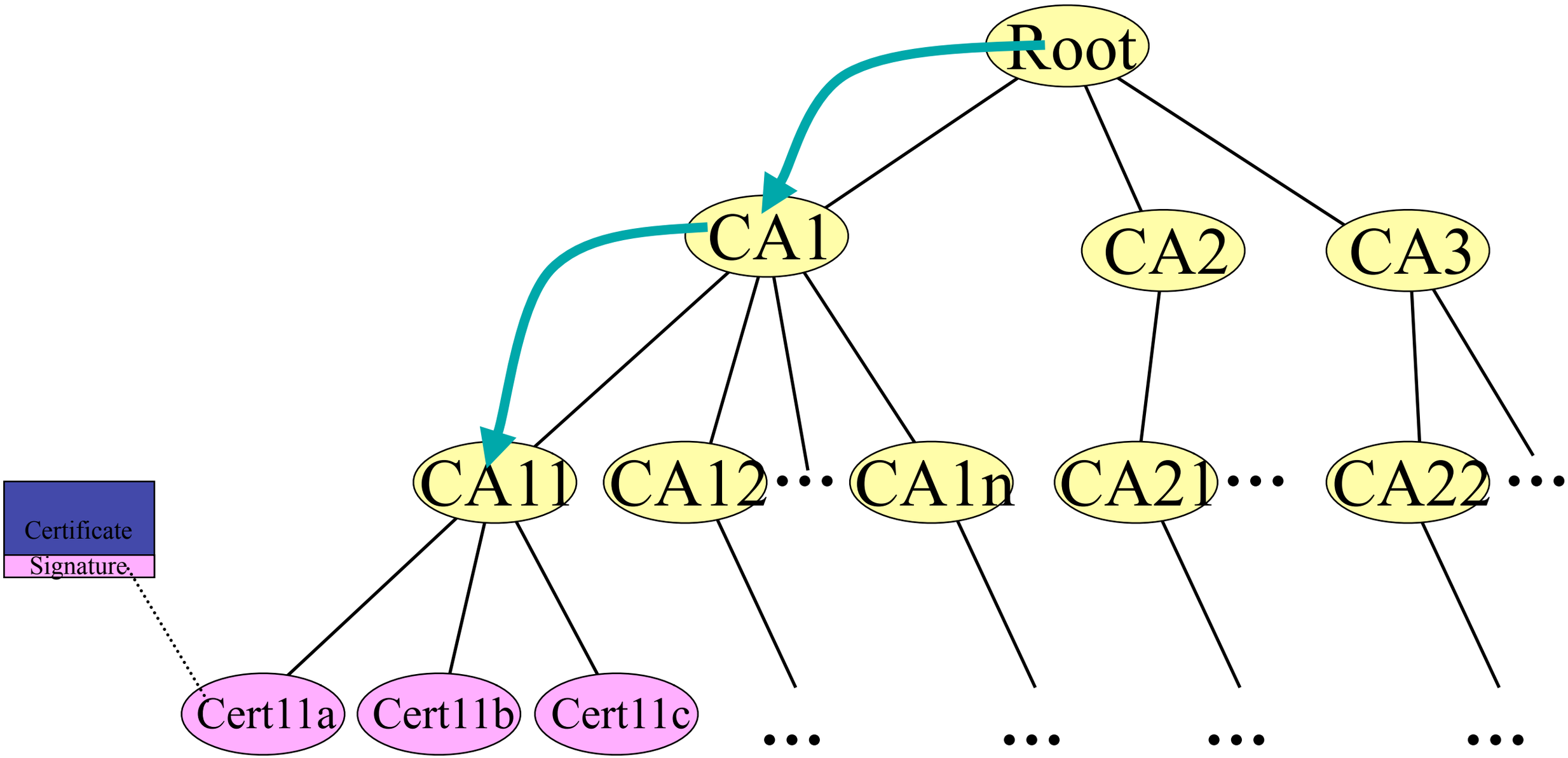


The Internet PKI?

- Rooted tree of CAs
- Cascading issuance
  - Any CA can issue cert
  - CAs issue certs for children



Certificate Validation

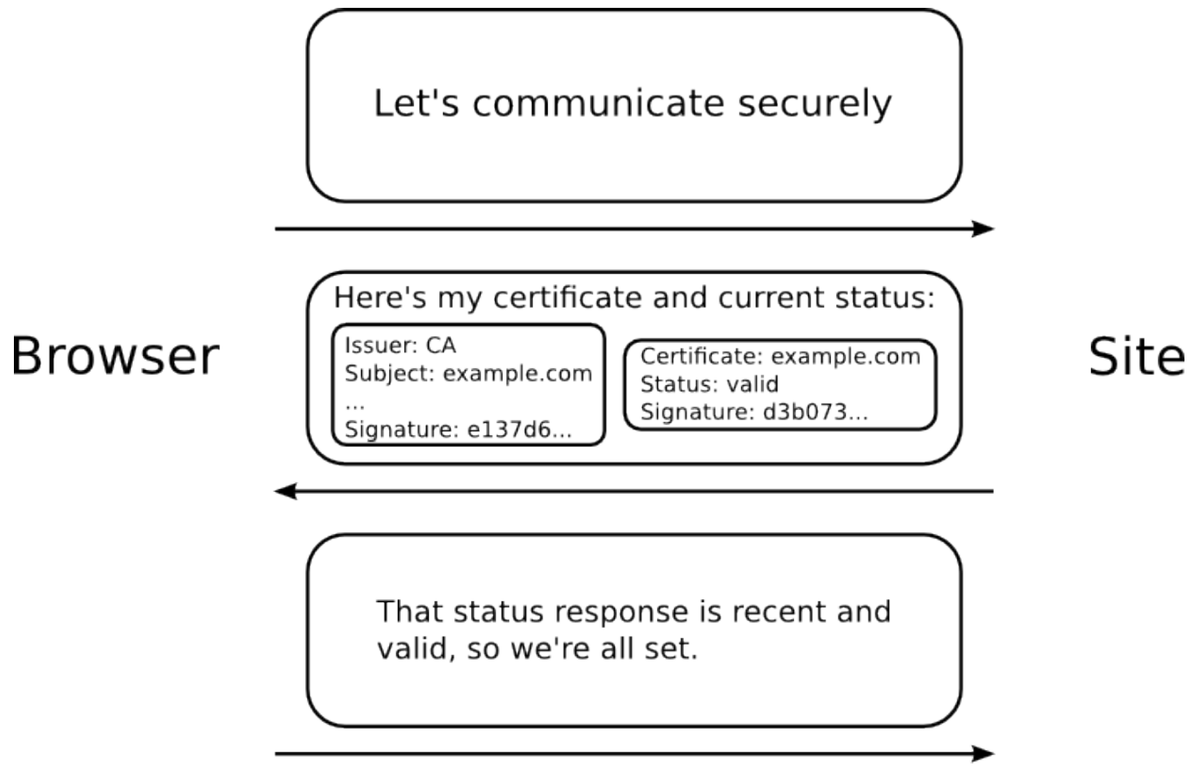


PKI and Revocation

- Certificate may be revoked before expiration
  - Lost private key
  - Compromised
  - Owner no longer authorized
- Revocation is hard …
  - The "anti-matter" problem
  - Verifiers need to check revocation state
    - Loses the advantage of off-line verification
  - Revocation state must be authenticated

Revocation Mechanisms

- Certificate revocation lists (CRL)
  - Periodically issued
  - Delta CRLs when CRLs get too large
- Online certificate revocation server
  - Answers revoked = yes/no for a particular certificate
    - Implemented by OCSP protocol
  - Disadvantages?
  - OCSP-stapling



Where's my PKI?

- Really talking about a full PKI (everyone has certs.)
- Why is that not a reality?
  - PKI was, like many security technologies, claimed to be a panacea
  - It was intended to solve a very hard problem: build trust on a global level
  - Running a CA -- "license to print money"

- Basic premise:
  - Assertion #1 - e-commerce does not need PKI
  - Assertion #2 - PKI needs e-commerce
- What are the problems?

Where's my PKI?

PennState

- Some of the problems with creating a per-user PKI?
  - Who has the private key? (Security of client hosts)
  - How do I manage my private key(s)? (Usability)
  - Which users is a CA an authority over? (Root of Trust)
  - How do users find a legit CA? (Trusted Path)
- Argument: We are trying to solve a painful problem: authenticating users.
  - What technical expectations can we make about users?

Burning question …

- Can we solve the PKI problem with better crypto?

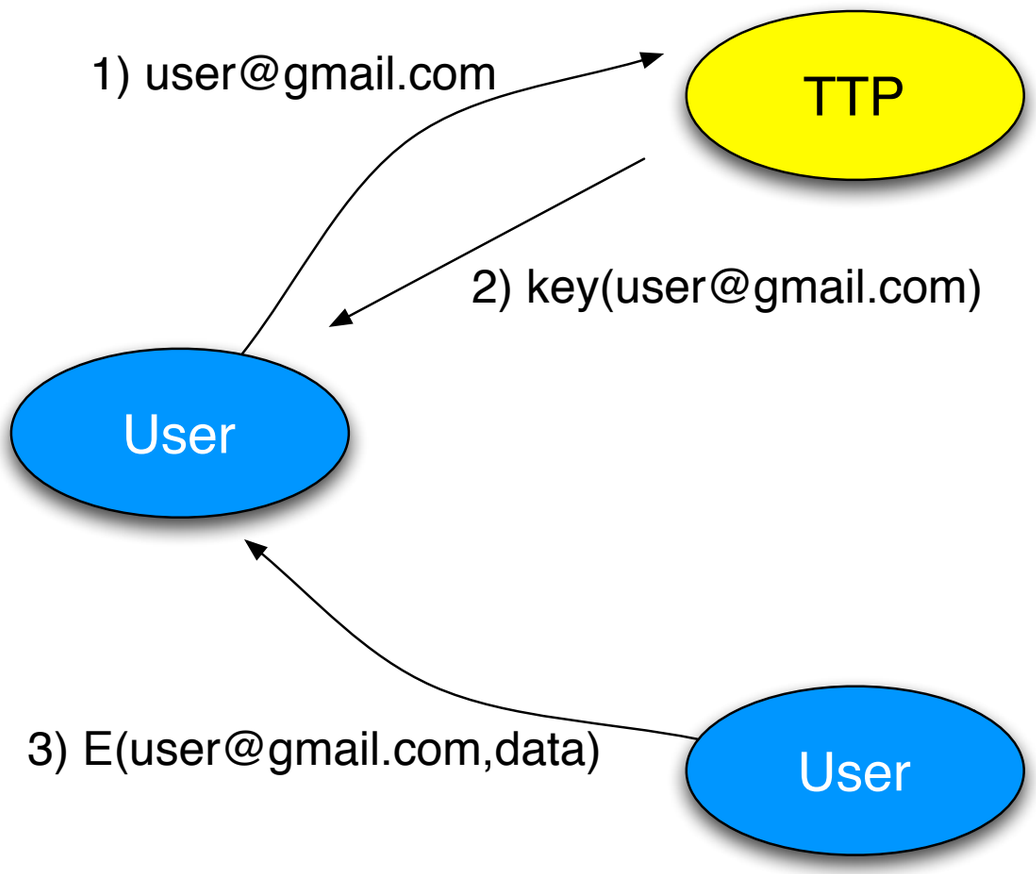
Identity Based Cryptography

PennState

- What if your email address was your public key?
  - E.g., E(jaeger@gmail.com, data) = ciphertext?
  - E.g., Verify(signature, jaeger@gmail.com)
- 1984 - Shamir asked for such a system, but it (largely) remained out of reach until Boneh/Franklin 2001
  - The public key is any arbitrary key
  - Based on "Weil pairings" -- a new cryptographic device with lots and lots of uses (IBE among them)
  - Interested readers should see: Identity based encryption from the Weil pairing, SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003.
- Advances from theory community, few systems

IBE System

- Functionally, you receive your private key from a *trusted third party* who is responsible for generating all keys in the system.
- Thereafter you (and others) can use the system as if you generated the private key yourself.
- Advantages
  - No public key distribution
  - No name binding problems (?)
  - Key space flexibility
  - Others?



Basic IBE Construction

PennState

- *Setup* (generate by TTP)

$$Global\ Parameters = G$$
$$Master\ Key = K_G$$

- *Extract* (by TTP for user, string "str")

$$Extract(G, K_G, Str) = K^-_{Str}$$

- *Encrypt* (for user)

$$E(G, Str, data) = ciphertext$$

- *Decrypt* (by user)

$$D(G, K^-_{Str}, ciphertext) = data$$

IBE Reality

- Many thought that IBE would lead to a revolution in public key system (solve PKI problems), it didn't.
- Why - IBE moves the problems around
  - Is there any TTP that everyone trusts?
  - String ambiguity is still a problem? (John Robinson?)
  - Revocation is still a problem (potentially worse)
- Fundamentally
  - IBE really does not solve the CA problem, as the TTP is fulfilling that role.
  - Having strings instead of obscure numbers does not get at the problems with PKI ...
  - Existence of certificates is not really the problem ...